

Remarks

This Amendment responds to the final Office Action (“the Action”) mailed July 27, 2007. Reconsideration of the application is respectfully requested in view of the following remarks. Claims 1-52 are pending in the application. No claims have been allowed. Claims 1, 17, 29, and 44 are independent.

Cited Art

U.S. Patent App. Pub No. 2002/0133639 to Breslau et al. (“Breslau”) is entitled “Method and System for Migrating an Object Between a Split Status and a Merged Status.”

U.S. Patent App. Pub No. 2004/0133556 to Wolczko et al. (“Wolczko”) is entitled “Method and Apparatus for Skewing A Bi-Directional Object Layout to Improve Cache Performance.”

Amendments

Editorial amendments have been made to claims 1, 17, 29, and 44. No new matter is added.

Claim Rejections under 35 USC § 102

The Action rejects claims 44-52 under 35 U.S.C. § 102(e) as being anticipated by Breslau. Applicants respectfully disagree and traverse the rejection. For a 102(e) rejection to be proper, the cited art must show each and every element as set forth in a claim. (See MPEP § 2131.01.) However, the cited art does not describe each and every element. Accordingly, Applicants request that the rejection be withdrawn. Claim 44 is independent.

Claim 44, as amended, recites, in part:

an analysis module for determining a coallocation solution based at least in part upon a temporal data access profile of a computer program, *the coallocation solution comprising one or more directions to allocate particular heap objects to particular heap memory arenas;*

...
an enforcement module for automatically enforcing the coallocation solution during execution of the altered computer program by *coallocating sets of heap objects in heap memory arenas according to the one or more directions of the coallocation solution.*

For example, the Application describes examples of coallocation solutions which are later enforced, at page 16:

In some embodiments, a profile-based analysis tool produces a cache-conscious coallocation of heap objects that participate in hot data streams, when possible. . . .

In some embodiments, an instrumentation tool replaces a program's heap allocation requests with calls to a run-time coallocation library to enforce the coallocation solution layout. . . .

[]The tool makes the following change to the second line:

buffer = (int *) comalloc_i (sizeof (int1));

where comalloc_i is a call to a particular routine that causes the allocation to be in the ith arena. *The tool enforces the coallocation solution in Figure 2C by replacing allocations at sites A, B, and C with calls to comalloc₁, and replacing allocations at other sites (including sites D and E) with calls to comalloc₀. Each comalloc_i routine manages a separate heap arena where consecutively allocated objects are coallocated.*

. . .

[Application, at page 16, lines 10-29.] The Application also describes an analysis module outputting coallocation solutions, which are then input into an enforcement module, at page 31:

The tool 1100 includes one or more analysis modules 1120 for analyzing the profile 1115 and determining one or more cache-conscious coallocation solutions 1125.

The analysis module(s) 1120 receive a profile 1115 such as a data reference trace for subsequent analysis. For example, the tool cmanal efficiently analyzes a context-free grammar representation for a profile to find hot data streams with their associated normalized heat value. The tool cmanal applies a coallocation algorithm (as described in the previous sections) to produce a collection of coallocation sets. The allocation context abstraction level (i.e., just the allocation site, or a calling context of some length l) is a tunable parameter, but may also be pre-defined. Alternatively, the analysis module(s) include other and/or additional modules.

Finally, the tool 1100 includes one or more enforcement modules 1130 for enforcing the cache-conscious coallocation solution(s) 1125 during execution of the program 1101.

[Application, at page 31, lines 1-13.] Finally, the Application provides many examples of such solutions, and the direction they provide, such as the list on page 26, lines 8-14.

Breslau does not produce a "coallocation solution comprising one or more directions to allocate particular heap objects to particular heap memory arenas" because it does not discuss any allocation of particular objects in particular heap memory arenas. Breslau is directed to "a method and system for migrating an object between a merged status having a single instance and a split status having multiple instances." [Breslau, at page 1, paragraph 0005.] Breslau's self-described motivation

is to solve problems such as load planning. [See, Breslau, at page 1, paragraph 0011.] Breslau also describes the need for load planning:

Objects are shared resources that may be invoked (i.e., by invoking their methods) by other objects throughout the object-oriented computer system. The load on an object (and the corresponding load on the execution environment it is instantiated within) will therefore vary with the periodicity of invocations of the object and complexity of the methods used within the object. *Careful system planning is required such that enough instances of any particular object are available to handle the presented load. . . .*

[Breslau, at page 1, paragraph 0008; emphasis added.] Thus, Breslau is directed to solving a problem of needing *enough instances* of objects to handle program loads. To solve this problem, Breslau “comprises a method of managing the object at runtime and includes identifying a request to migrate the object between a split status and a merged status.” [Breslau, at page 1, paragraph 0012.] Breslau does not discuss the usage of object splitting or merging to address cache performance, nor does it ever discuss caching.

In contrast to Breslau’s stated goals, Applicants note that Breslau does not discuss a “coallocation solution comprising one or more directions to allocate particular heap objects to particular heap memory arena” as recited in claim 44. In its rejection of the “analysis module” language of Breslau, the Action cites to paragraph 0015 of Breslau. [Action, at § 5, page 2.] However, this passage describes only “presevering the state of [an] object” (Breslau, at page 1, paragraph 0015) and does not discuss allocation of heap objects in heap memory.

The Action, in both its rejection of the “enforcement module” language, as well as in its Response to Arguments section, cites to paragraphs 0036-0038 of Breslau. [Action, at § 5, page 2, and §8, pages 18-19.] However, as Applicants have previously pointed out, in this passage, Breslau discloses only a process for “splitting” of an object into multiple objects in order to increase processing capacity. [See, Breslau, at page 3, paragraph 0032.] The passage does not describe object allocation; in other words, the passage does not describe *where* the objects are allocated in heap memory. In fact, when Breslau later describes the instantiation of objects, it makes clear that its methods are unconcerned with the particulars of allocation:

A next step in the process includes instantiating multiple instances of the object being split into multiple execution environments (STEP 67). For example, in FIG. 3, multiple instances of “object B” (e.g., objects B1 61, B2 63 and B3 65) are loaded into

the workstations. *The creation of an object within an execution environment is conventional.*

[Breslau, at page 4, paragraph 0039.] Thus, Breslau indicates that it does not describe particular allocation of objects, let alone “a coallocation solution comprising one or more directions to allocate particular heap objects to particular heap memory arenas.”

Breslau’s discussion of “object splitting” does not read on “an analysis module for determining a coallocation solution” where “the coallocation solution compris[es] one or more directions to allocate particular heap objects to particular heap memory arenas” as Breslau does not describe performing an analysis to determine a solution. In a section cited in various rejections, Breslau describes the receipt of a “‘split’ request,” which causes it to perform a process for splitting an object:

The process begins with the generation and/or receipt of a "split" request (STEP 61). This may be generated by another program or object executing within the object-oriented computer system. As one example, the split request may be generated by a performance monitoring program that has determined an overload condition for a "merged" object within an execution environment.

[Breslau, at page 3, paragraph 0036.] As Breslau notes, this “split request” describes only a realization that *more objects are needed*. At best this analysis is a simple yes/no type of indication that reports if there are not enough objects in memory. This simple analysis cannot read on an analysis that determines a “coallocation solution” comprising “directions to allocate particular heap objects to particular heap memory arenas” as recited in claim 44. Indeed, Breslau cannot describe such an analysis, as Breslau does not know about particulars of object allocation, as discussed above.

For at least these reasons, Breslau does not teach or suggest the above-quoted language of claim 44 and thus, Breslau does not describe each and every element as set forth in claim 44. The rejection of claim 44 over Breslau is thus improper. Applicants respectfully note that claim 44 is allowable and request that claim 44, as well as dependent claims 45-52, be allowed.

Patentability of Claims 1-43 Under 35 USC § 103(a)

The Action rejects claims 1-43 under 35 U.S.C § 103(a) as unpatentable over Breslau in view of Wolczko. To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference

teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. [See, MPEP § 2142.]

Claim 1

Claim 1 recites, in part:

analyzing the one or more hot data streams and the temporal data reference profile to determine a coallocation solution for allocations in heap memory, *the coallocation solution comprising direction to allocate particular heap objects to particular heap memory arenas;*

[Emphasis added.] In its rejection of the above-quoted language of claim 1, the Action cites to the paragraphs 0032-0035 and 0053-0055 of Breslau. [See, Action at § 7, page 5.] For reasons similar to those discussed above with respect to claim 44, paragraphs 0032-0035 of Breslau does not teach or suggest “analyzing the one or more hot data streams and the temporal data reference profile to determine a . . . coallocation solution comprising direction to allocate particular heap objects to particular heap memory arenas” as is recited in claim 1. Applicants furthermore do not find such disclosure in paragraphs 0053-0055 of Breslau, which describes management of already-split objects, such as decisions about invocation of objects and informing objects of split or merged status. [See, Breslau, at page 5, paragraph 0053-0055.] Furthermore, Applicants do not find such disclosure in Wolczko, which is directed to a particular method of skewing object layout during operation. (See, Wolczko, at Abstract) and which does not appear to teach or suggest a “coallocation solution” as recited in claim 1.

For at least these reasons, the combination of Breslau and Wolczko does not teach or suggest every element of claim 1. Claim 1 is thus allowable and applicants request its allowance.

Claim 17

Claim 17 recites, in part:

determining a coallocation solution based at least in part upon the profile, wherein the coallocation solution *comprises one or more directions to allocate particular heap objects to particular heap memory arenas to increase locality of object fields in a layout in memory to improve cache performance;*

[Emphasis added.] In its rejection of the above-quoted language of claim 17, the Action cites to the paragraphs 0032-0035 and 0051-0055 of Breslau. [See, Action at § 7, page 5.] For reasons similar to

those discussed above with respect to claims 44 and 1, Breslau does not teach or suggest the above-quoted language of claim 17. Furthermore, Applicants do not find such disclosure in Wolczko. For at least these reasons, the combination of Breslau and Wolczko does not teach or suggest every element of claim 17. Claim 17 is thus allowable and applicants request its allowance.

Claims 29

Claim 29 recites, in part:

analyzing the one or more data access patterns and the temporal data access profile to determine a coallocation solution for allocations in memory, *the coallocation solution comprising one or more directions to allocate particular heap objects to particular heap memory arenas;*

[Emphasis added.] In its rejection of the above-quoted language of claim 29, the Action cites to the paragraphs 0032-0035 and 0053-0055 of Breslau. [See, Action at § 7, page 5.] For reasons similar to those discussed above with respect to claims 44 and 1, Breslau does not teach or suggest the above-quoted language of claim 29. Furthermore, Applicants do not find such disclosure in Wolczko. For at least these reasons, the combination of Breslau and Wolczko does not teach or suggest every element of claim 29. Claim 29 is thus allowable and applicants request its allowance.

Interview Request

If the claims are not found by the Examiner to be allowable, the Examiner is requested to call the undersigned attorney to set up an interview to discuss this application.

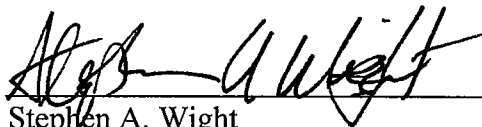
Conclusion

The claims in their present form should be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

By 
Stephen A. Wight
Registration No. 37,759